Improvement of Data Integrity and Data Dynamics for Data Storage Security in Cloud Computing

Poonam M. Pardeshi and Bharat Tidke

Abstract Cloud stands today as an emerging standard, however, data outsourcing paradigm is main security concern in cloud. To make sure that the data stored on the cloud is safe, frequent data integrity checking is imperative. This work considers the problem of data integrity in cloud storage and makes use of Dynamic Merkle Hash Tree (DMHT) along with AES and SHA-1 algorithms to solve the same. RSA algorithm has been used by many previously developed systems; the proposed work makes use of AES which leads to performance improvement. The work also makes use of the concept of Third Party Auditor (TPA) to achieve Public Auditing. In case of corruption of data or data loss, the proposed work promises to recover the lost data with the help of a backup system. In order to support dynamic data operations, the Merkle Tree is made dynamic by making use of relative index. Further, to save the communication bandwidth and cost, block level recovery is made instead of recovery of entire file. On comparison with previous systems, the proposed system shows reduction in server computation time. The proposed work thus aims at improving and maintaining data integrity at untrusted server, supports dynamic data operations and makes recovery possible by providing a recovery system.

Keywords Advanced encryption standard · Cloud computing · Data dynamics · Merkle hash tree · Public auditability · Recovery system · Secured hash algorithm, third party auditor

P.M. Pardeshi (✉) · B. Tidke
Department of Computer Engineering, Flora Institute of Technology,
University of Pune, Pune, Maharashtra, India
e-mail: ppardeshi31@gmail.com

B. Tidke
e-mail: batidke@gmail.com

J.K. Mandal et al. (eds.), *Information Systems Design and Intelligent Applications*,
Advances in Intelligent Systems and Computing 339,
DOI 10.1007/978-81-322-2250-7_27

1 Introduction

Cloud computing has become an emerging and highly appealing trend due to its innumerable benefits. With its massively large storage centers, low cost, high scalability, flexibility in access points and less client system resources utilization, it has successfully diverted many clients towards it. This increase in the number of cloud users along with their sensitive data renders the data outsourcing paradigm in cloud as security demanding area. The dilemma becomes graver when the question arises for the security of confidential data. If the server or Cloud Service Provider (CSP) is untrusted, client may lose important data [1–14, 15–18]. For example, less frequently accessed data can be deleted by server to save storage space on cloud, for personal benefits server may try to hide data errors at the time of Byzantine failures. The cloud thus has many security concerns related to its storage [2–8, 12–14]. Previously, much work has been done on improving the storage security in cloud by verifying the data integrity. All of these works fall under either Private Auditing or Public Auditing. In Private Auditing only client has the right to verify its own data [5, 6] whereas in Public Auditing client can delegate the authority of data verification to a third party on its behalf [4, 9, 11, 13–15, 18]. This third party is referred as a Third Party Auditor (TPA). This paper focuses on improving the data integrity by using DMHT, AES and SHA-1 algorithms. In proposed system, server is assumed to be an untrusted entity, so, the data to be stored on it is encrypted priori using AES-128 algorithm. If the key size of AES is increased from 128 to 192, power and time consumption increases by 8 % and an increase of 16 % is caused by 256 bits key [19, 20], hence, AES-128 has been used in the proposed work both for signing the root of the DMHT and data encryption. The work also assures data availability and recoverability at the time of unpleasant situations at the server such as a server crash in which integrity of data is lost, by providing a backup and recovery system. DMHT has been used to make dynamic operations possible.

2 Background Theory

2.1 Third Party Auditor (TPA)

The TPA is an entity which is equipped with capabilities, knowledge, expertise and skills that client does not possess. It works on behalf of client and is externally allotted by client itself to verify integrity of its data. In other words, it reduces the overhead of client and client no longer needs to do the job of verification on its own.

Cloud Storage Architecture:
Figure 1 shows cloud storage architecture. It has three network entities viz. client, CSP and TPA present in it. Client stores data on cloud server, CSP is the service provider where data is stored and TPA is responsible for auditing the stored data.

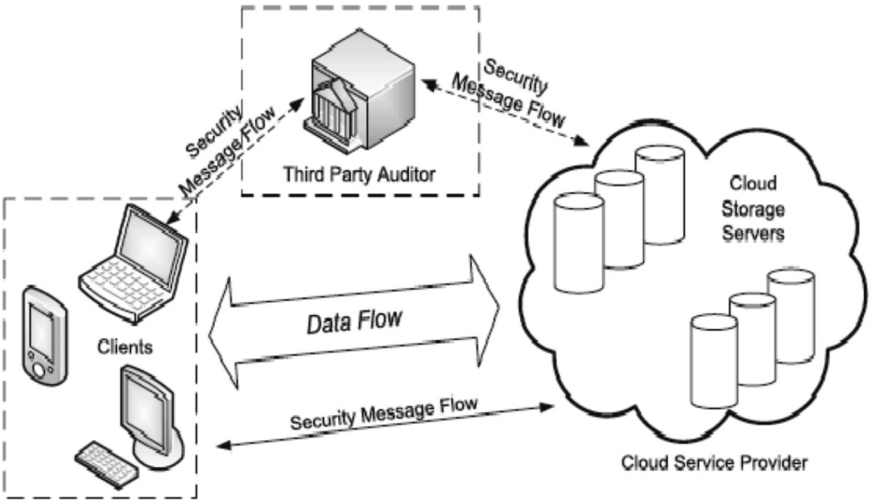

Fig. 1 Cloud storage architecture [1]

2.2 Merkle Hash Tree (MHT) and Dynamic Merkle Hash Tree (DMHT)

A Merkle Hash Tree is a used widely for authentication of file blocks by cryptographic structures. Its use greatly reduces server computation time [13]. Its construction takes place in similar fashion as a normal binary tree. However, in this paper, we make use of a DMHT instead of a simple MHT to make dynamic data operations possible. Each node of a DMHT has two auxiliary information viz. a hash value and a relative index unlike a static MHT whose leaf nodes has only hash value [11]. Relative index is a term used for extra data filed carried by each node of DMHT, which is used to indicate number of leaf nodes in the subtree of a node. An example of a MHT is shown in Fig. 2, to make it dynamic, we make use of relative index. So, if there exists a node 'R' with 'a' as left child and 'b' as right child then the information carried by node 'a' will be (ha, na), node 'b' will be (hb, nb)and relative index of root 'R' will be nr = na + nb. In Fig. 2, relative index of node 'a' is 2, 'b' is also 2 and that of 'r' is 4. To present this concept more clearly, we represent a DMHT in Fig. 3.

Organization. Section 3 presents a survey on various systems developed for providing storage security in cloud. Section 4 describes proposed model along with implementation details. Section 5 shows performance analysis and Sect. 6 provides conclusion and future work.

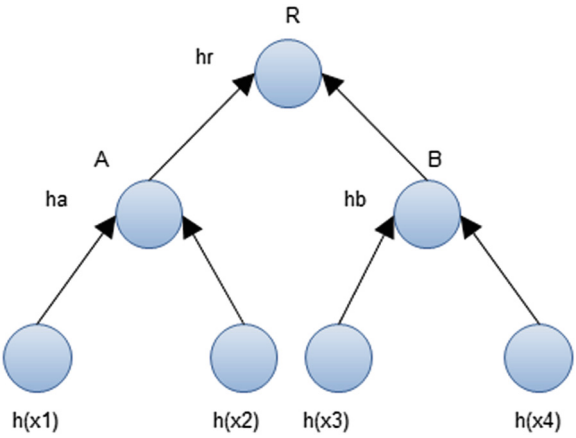

Fig. 2 Merkle hash tree

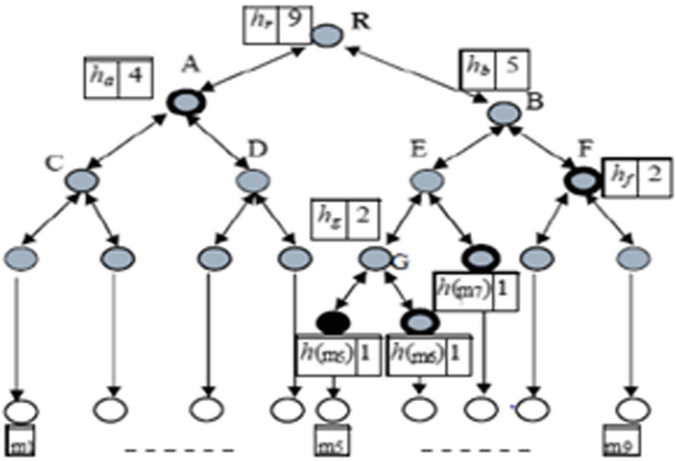

Fig. 3 A dynamic Merkle hash tree [11]

3 Literature Review

A lot of work has been done in the storage security area of cloud out of which most work has focused on integrity verification of data stored in cloud. Deswarte et al. [1], makes use of RSA based hash function for file verification. Client can generate multiple challenges using same metadata in this scheme. The computational complexity at the server adds to the limitation of this scheme. A technique proposed by Schwarz and Miller [3] makes use of algebraic signature. In this, a function is used to fingerprint the file block. The computation complexity at client and server side takes place at the cost of linear combination of file blocks, also there are issues related to the security of this

Table 1 Comparison between different systems

Scheme	Ref. no. attributes					
	[4] G. Ateniese et al	[5] A. Juels et al	[6] G. Ateniese et al	[9] C. Wang et al	[20] S. Zhong et al	Proposed system (AESSS)
Privacy preserving	No	Yes	No	No	Yes	Yes
Unbound No. of queries	Yes	No	No	Yes	Yes	Yes
Public verifiability	Yes	No	No	Yes	Yes	Yes
Use of TPA	No	No	No	Yes	No	Yes
Recoverability	No	Yes	No	No	No	Yes
Dynamic operations	No	No	Yes	Yes	Yes	Yes
Untrusted server	Yes	Yes	Yes	Yes	Yes	Yes

scheme. Ateniese [4] were the first who considered Public Auditing for ensuring possession of files at untrusted servers. The Provable Data Possession (PDP) model supports large data sets in widely-distributed storage systems and is provably-secure for remote data checking. This scheme imposes an overhead of generating metadata on client and provides no support for dynamic auditing. Juels [5], "Proofs of Retrievability" (POR), focuses on static archival of large files. Spot checking and error correcting codes are used in this scheme to ensure data possession and retrievability. Drawback: it cannot be used for public databases and is suitable only for confidential data. Dynamic updation is not possible because of the sentinel nodes, public auditing is not supported by the scheme. Scalable and Efficient Provable Data Possession (S-PDP and E-PDP) protocols [6] makes contribution to the work of Ateniese [4]. It is dynamic version of PDP scheme and relies only on efficient symmetric-key operations. It makes use of less storage space by reducing the size of challenge and response blocks and uses less bandwidth. Shortcomings: number of queries that can be answered are fixed priori, partially dynamic scheme as block insertion is not supported. The scheme proposed by Erway et al. [8] is a dynamic auditing protocol which supports public auditing. It supports data dynamics via general data operation such as block insertion, deletion and block modification. However, the scheme may leak data content to the auditor as it needs linear combination of file blocks to be sent to the auditor for verification. Also, the efficiency of this scheme is not clear. Table 1 shows comparison of different existing systems with proposed one.

4 Proposed Work

Design

Figure 4 represents data flow diagram of AES based Storage Security System which has three network entities viz. client-who stores data on cloud, CSP-generates the proof for data stored in it and TPA-an entity that performs proof verification. Backup server serves the purpose of file recovery.

Notations: Esk—Secret key encryption, F—File stored at untrusted cloud server, x—File block, T—Tag (signature), Φ—Set of tags

4.1 The Security Model

The proposed storage security model has two phases: The setup phase and the Integrity verification phase.

4.1.1 The Setup Phase

In this phase, client generates a file F = {x1, x2, …, xn} which is a collection on n number of file blocks The setup phase has five steps. In first step, for each file block, a signature is generated using secret key, given as Ti = Esk(H(xi)), where xi is the ith block of file. In second step all the signatures are collected together to make a signature set called set of tags, represented as Φ = {Ti}. Then DMHT is constructed and root of tree is signed using secret key as sigsk (H(R)). In last step, client advertises {F, Φ, sigsk (H(R))} to the server and deletes F and Sigsk(H(R)) from local storage Fig. 5.

4.1.2 Integrity Verification Phase

The integrity verification process, in Fig. 6, is initiated by client by sending an auditing request consisting of some metadata such as FileId and ClientId, to TPA for a particular file. The TPA then generates a challenge and sends it to the server, for which the server generates a proof. The proof contains the root and signature of the DMHT generated for that particular file. The proof is then sent to the TPA which performs integrity verification. Proof verification is done in two stages; firstly signature of the root is checked for file authentication. For this, the output is true if the signature matches with the one stored during file upload, otherwise false. If true, then the value of the root is checked. If the value of the root is same as that stored

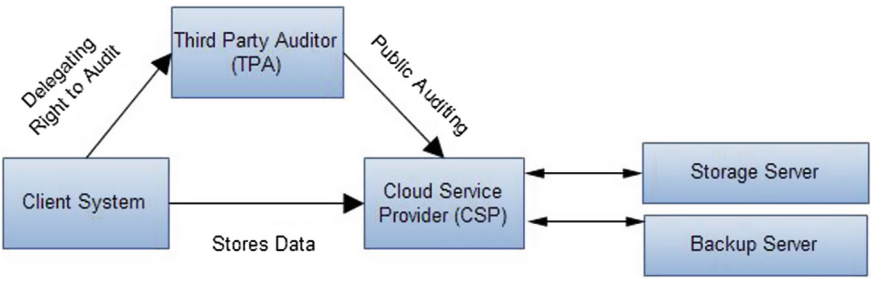

Fig. 4 Data flow in AES based storage security system

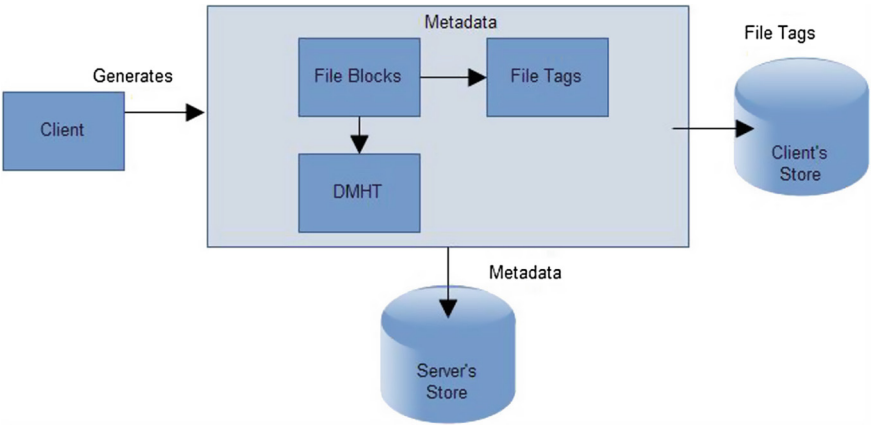

Fig. 5 Pre-processing the file blocks

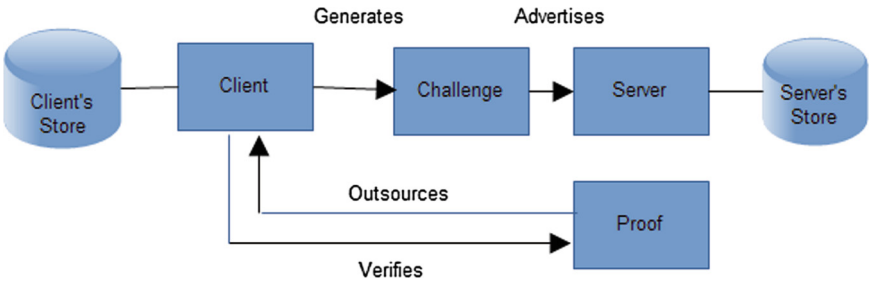

Fig. 6 Integrity checking process flow [14]

previously, then the file is integrated otherwise some part of the file is modified or lost and the file has lost its integrity. Any changes made to any part of the file are reflected in the root of the tree and so for integrity verification, checking only the value of the root is sufficient. Client is notified about the file's condition after verification. The process is Privacy-Preserving as TPA views only the file tags for verification and not the actual data. When a file is found to be infected, block level checking is done to find out particularly which block is infected.

4.2 The Recovery System

Users can store a copy of their file in the backup section so that it can be recovered if server undergoes any unpleasant situation such as server crash or link failure or loss or corruption of original file data. Here, block level recovery is done by

fetching exactly the infected block instead of entire file from backup server. This reductions required bandwidth. The recovery system makes data availability possible and hence adds to the plus points of the proposed system.

4.3 Dynamic Operations

4.3.1 Method for searching (i-th leaf node)

To perform dynamic operations, searching algorithm is necessary. Firstly i is compared with index (n) of root node. If i is greater than the root node's index then False is emitted, else, we consider k = i and (ha, na) be the left subtree and (hb, nb) is right subtree. We now compare k with the relative index of the left child. If $k \leq na$ then k lies in left subtree otherwise in right subtree. If it lies in right subtree, let $k = k - na$ and use this algorithm to find the node right subtree. This procedure is repeated until k = 1 i.e. a leaf node is reached.

1. Insertion
   When a block say x* has to be inserted after a block xi—ith block, signature T is generated for this block by using secret key. An update request is then constructed as update = (I, i, x*, T*) and sent to server. Server executes update operation and for this, it follows the steps as: it stores the block x* and leaf node h(H(x*)). (ii) It finds h(H(xi)) in DMHT, reserve Ωi and then inserts leaf node h (H(x*)) after i-th node. A new internal node is added to the tree with relative index as 2 and information of all nodes which fall between this node and root node are modified by recalculating their hashes and relative index. A new root node is generated based on the changes made.
2. Data Deletion and Modification
   Data deletion has similar process and is just the opposite of data insertion operation. Based on node searching algorithm, the required node is searched and then deleted. After deleting it the same procedure is followed as in data insertion. In data modification, the data is replaced and so the structure of the tree remains the same. The procedure followed is same as that in the data insertion.

5 Performance Analysis

5.1 Verification Time for Different Number of File Blocks

Figure 7 presents a graph for time taken for verification of different number of file blocks. Verification time varies according to variation in number of infected blocks. As observed form Fig. 7, time required for verification of a file when no block is

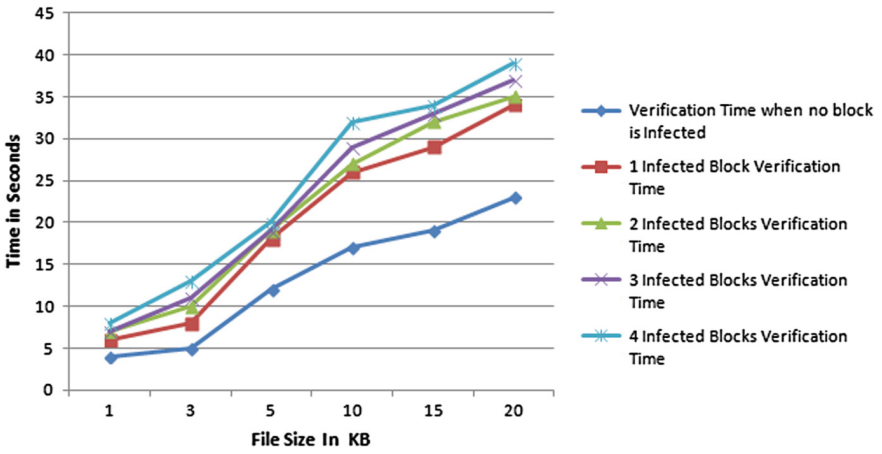

Fig. 7 Comparison of verification time for different number of file blocks

Table 2 Comparison of verification time for different number of file blocks

Sr. No.	File size in MB	Verification time in milliseconds when				
		No block infected	1 Block infected	2 Blocks infected	3 Blocks infected	4 Blocks infected
1	1	4	6	7	7	8
2	3	5	8	10	11	13
3	5	12	18	19	19	20
4	10	17	26	27	29	32
5	15	19	29	32	33	34
6	20	23	34	35	37	39

infected is least while this time increases gradually with the increment in number of infected blocks in a file. For example, for a file size of 3 MB, it takes 5 ms to verify the integrity of file if it is not infected, whereas, when its 1 block is infected it needs 8 ms, for 2 infected blocks 10 ms, for 3 infected blocks 11 ms and for 4 infected blocks it takes 13 ms. Table 2 gives a detailed view of graph represented in Fig. 7.

5.2 Server Computation Time

The graph in Fig. 8 shows that server computation time taken by AES based Storage Security System is less as compared to RSA based security system. For example for a file of size 90 kb, AESSS takes 1.3 s while RSA bases system takes nearly 3.8 s for server computation. Thus, the proposed system's server computation proves to be much less than the other systems.

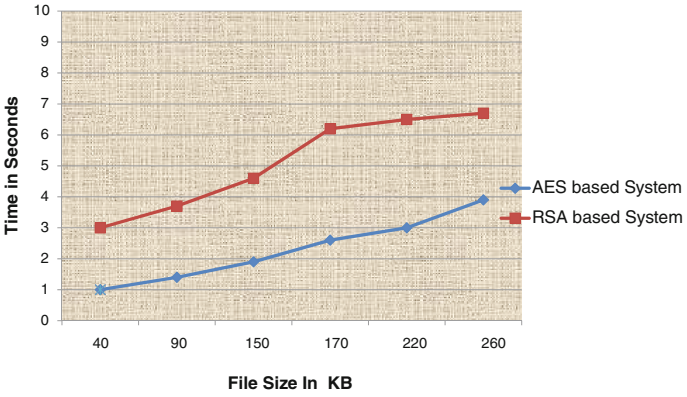

Fig. 8 Server computation time comparison

6 Conclusion and Future Scope

The proposed system ensures user data security at untrusted cloud server by frequent integrity checking of stored data. Use of AES algorithm instead of RSA for signature generation and encryption makes the scheme more secured and efficient. The system supports Public Auditing with the help of TPA and relieves its users from the overhead of integrity verification. Use of tags instead of actual data blocks for verification makes the auditing process time efficient and Privacy Preserving. The system supports dynamic data operations by constructing DMHT. The backup and recovery server takes care of availability of data during unpleasant situations at the server. Block level recovery highly reduces the communication cost and time for recovery.

References

1. Deswarte, Y., Quisquater, J., Saidane, A.: Remote integrity checking. In: Proceedings of Conference on Integrity and Internal Control in Information Systems (IICIS'03), Nov 2003
2. Sebe, F., Domingo-Ferrer, J., Martinez-Balleste, A., Deswarte, Y., Quisquater, J.-J.: Efficient remote data possession checking in critical information infrastructures. IEEE Trans. Knowl. Data Eng. **20**(8), 1034–1038 (2008)
3. Schwarz, T., Miller, E.L.: Store, forget and check: using algebraic signatures to check remotely administered storage. In: Proceedings of ICDCS '06. IEEE Computer Society (2006)
4. Ateniese, G.: Provable data possession at untrusted stores. In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS'07) (2007)
5. Juels, A.: Pors: proofs of retrievability for large files. In: Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07), pp. 584–597 (2007)
6. Ateniese, G.: Scalable and efficient provable data possession. In: Proceedings of the 4th International Conference on Security and Privacy in Communication Networks (SecureComm '08) (2008)

7. Xie, M., Wang, H., Yin, J., Meng, X.: Integrity auditing of outsourced data. In: Proceedings of the 33rd International Conference on Very Large Databases (VLDB), pp. 782–793 (2007)
8. Erway, C., Kuocu, A., Pamanthou, C., R.Tamassia: Dynamic Provable Data Possession. In: Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS'09) (2009)
9. Wang, C.: Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst. **22**(5), May 2011
10. Wang, C., Wang, Q., Ren, K., Lou, W.: Ensuring dynamic data storage security in cloud computing. In: Procedings of the 17th International Workshop Quality of Service (IWQos'09) (2009)
11. Chen, L., Chen, H.: Ensuring dyanmic data integrity with public auditing for cloud storage. In: Proceedins of International Conference on Computer Science and Service System (ICSSS' 2012) (2012)
12. Kunfam, L.M.: Data Security in the world of cloud computing. IEEE Secur. Priv. **7**(4), 61–64 (2009)
13. Venkatesh, M.: Improving Public Auditability. Data Possession in Data Storage Security for Cloud Computing, ICRTIT-IEEE (2012)
14. Hao, Z., Yu, N.: A multiple-replica remote data possession checking protocol with public verifiability. In: Proceedings of the 2nd International Data, Privacy and E-Com Symposium (ISDPE '10) (2010)
15. Zhou, M., Zhang, R., Xie, W., Qian, W., Zhou, A.: Security and privacy in cloud computing: a survey. In: Sixth International Conference on Semantics, Knowledge and Grids (2010)
16. Sravan Kumar R., Saxena, A.: Data integrity proofs in cloud storage, 978-1-4244-8953-4/11/ $26.00 ©, 2011 IEEE
17. Varalakshmi, P.: Integrity checking for cloud environment using encryption algorithm, 978-1-4673-1601-9/12/$31.00 ©, 2012 IEEE
18. Hao, Z., Zhong, S., Yu, N.: A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. IEEE Trans. Knowl. Data Eng. **23**(9) (2011)
19. Elminaam, D.S.A., Kader, H.M.A., Hadhoud, M.M.: Performance evaluation of symmetric encryption algorithms. IJCSNS Int. J. Comput. Sci. Netw. Secur. **8**(12), 280–286 (2008)
20. Singh, S.P., Maini, R.: Comparison of data encryption algorithms. Int. J. Comput. Sci. Comm. (IJCSC) **2**(1), 125–127 (2011)